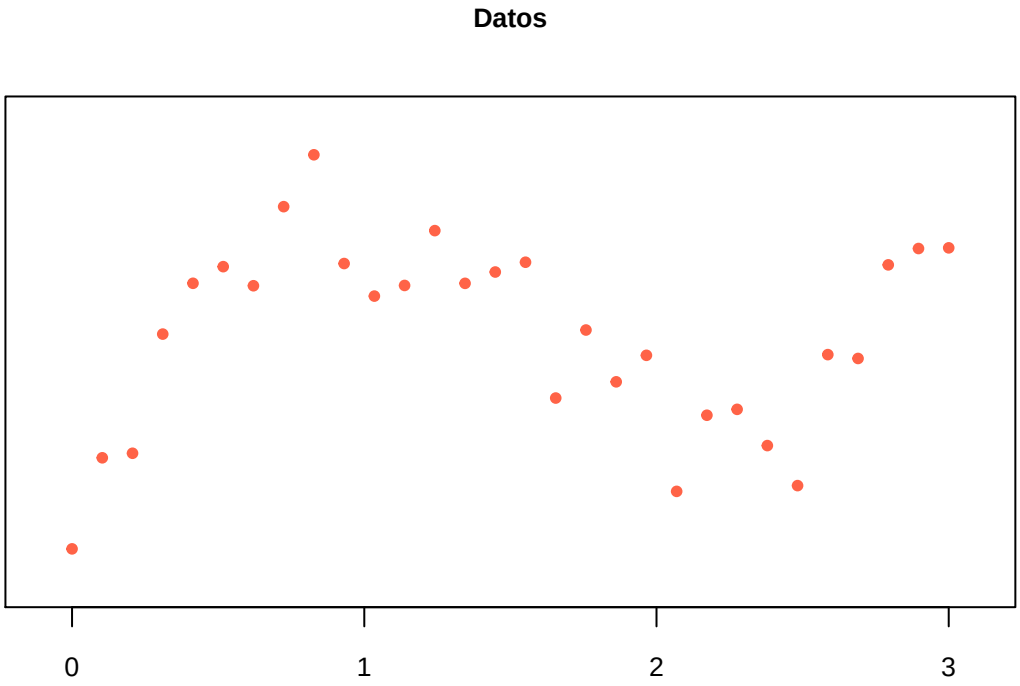
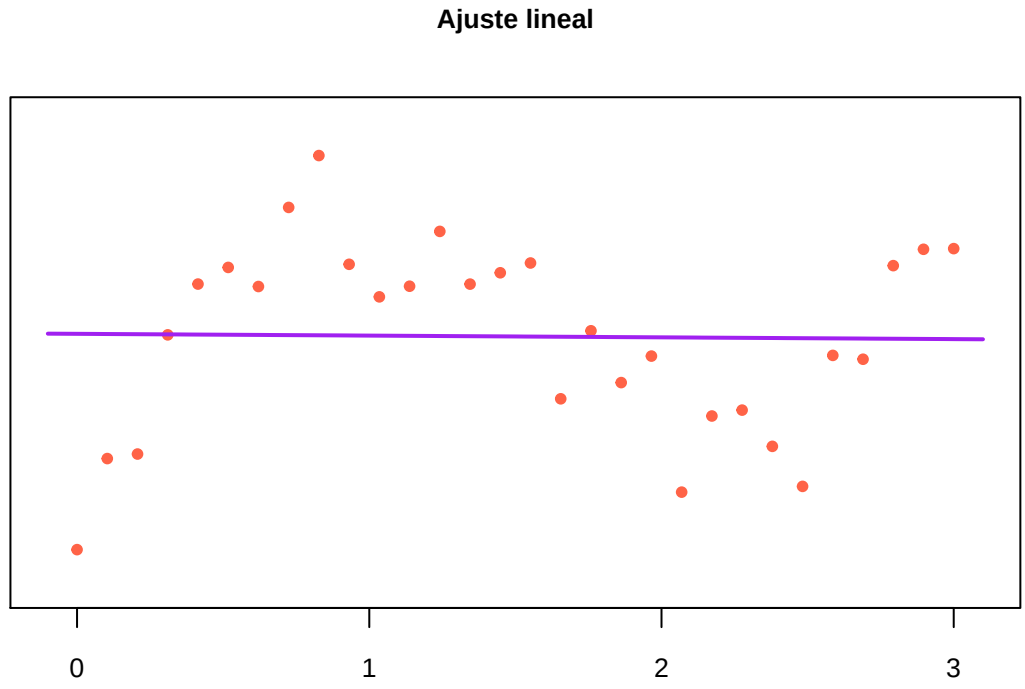
# Splines Cúbicos, Métodos Estadísticos I, 2017

Irving Gomez & Cricelio Montesinos 25/04/2017

```
set.seed(79015)
n   = 30
x   = seq(0, 3,length=n)
y   = 3*x-3*x^2+x^3-0.1*x^4+rnorm(n,0,0.15)
yr  = range(y)
yl  = yr+0.10*c(-1,1)*(yr[2]-yr[1])
plot(x, y, xlab="", ylab="", mgp=c(1.5,0.5,0), col="tomato", yaxt="n",
     xaxt="n", cex.axis=0.7, main="Datos", cex.main=0.8, pch=20,
     ylim=yl, xlim=c(-0.1,3.1))
axis(1,at=0:3,cex.axis=0.8)
```



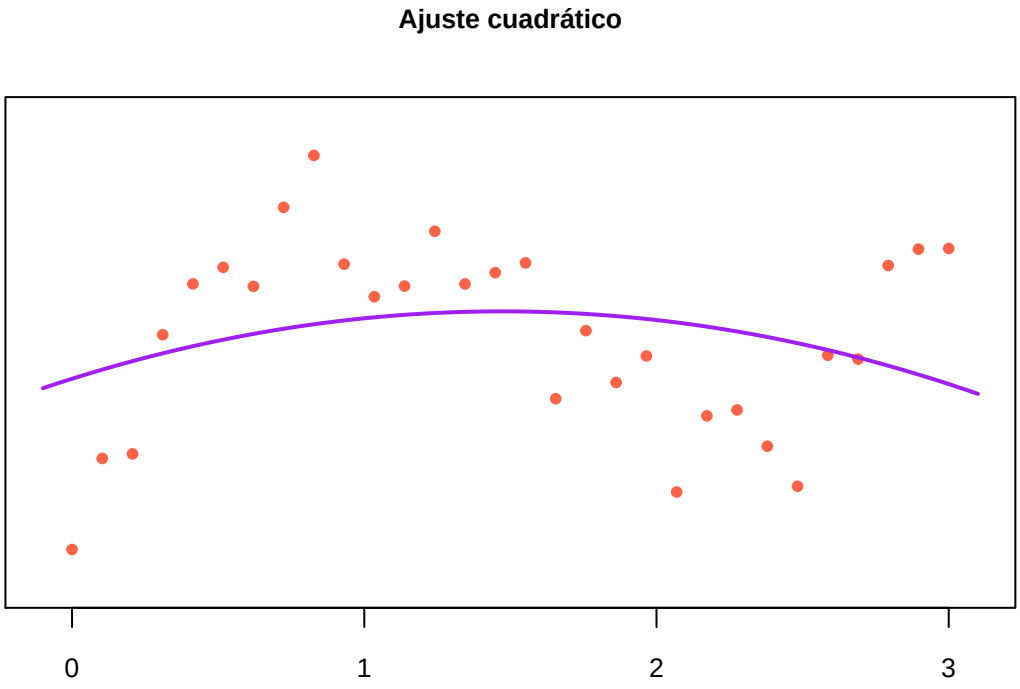
```r
modlin = lm(y~x)
betas = modlin$coefficients
plot(x, y, xlab="", ylab="", mgp=c(1.5,0.5,0), col="tomato", yaxt="n",
     xaxt="n", cex.axis=0.7, main="Ajuste lineal", cex.main=0.8, pch=20,
     ylim=yl, xlim=c(-0.1,3.1))
axis(1,at=0:3,cex.axis=0.8)
curve(betas[1]+betas[2]*x, add=T, col="purple", lwd=2)
```



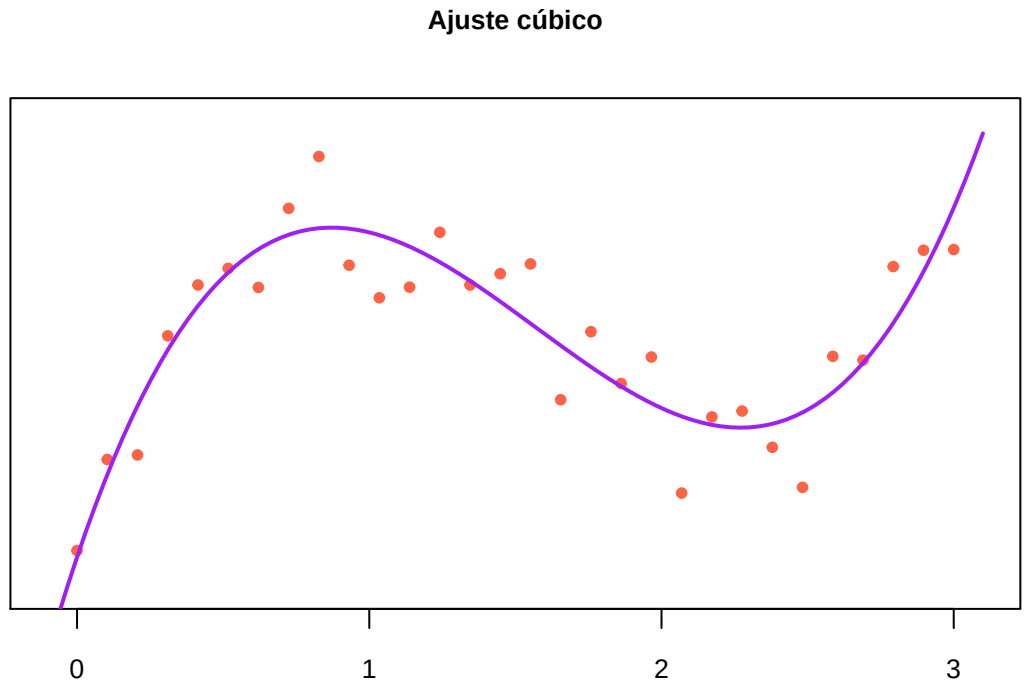
El ajuste no es nada satisfactorio.

¿Qué pasa si ajustamos polinomios?

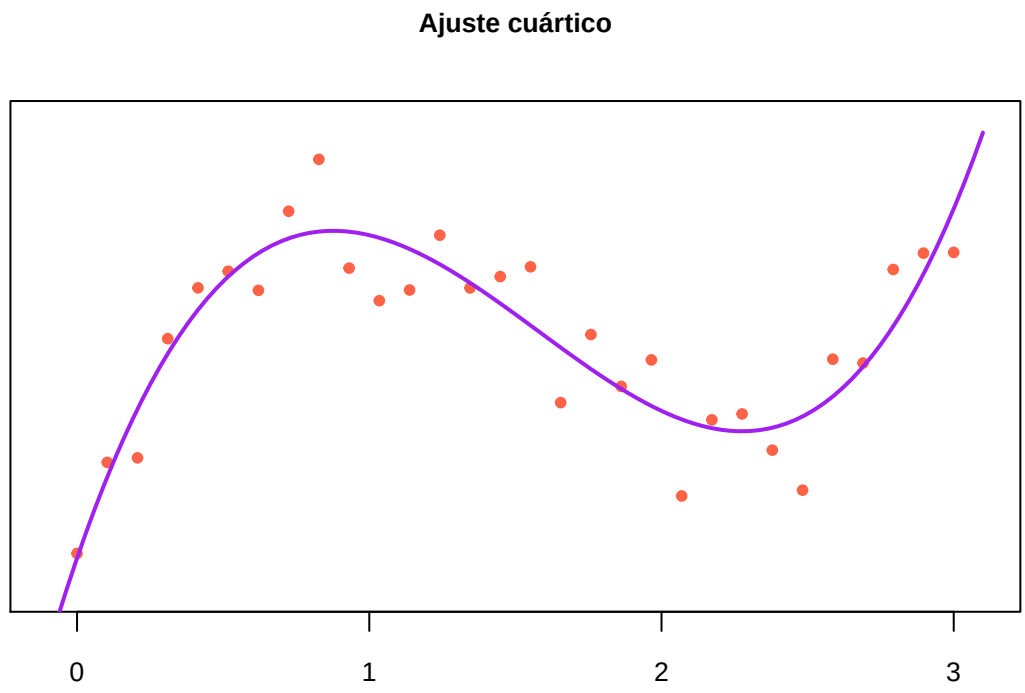
```
x2 = x^2
modlin = lm(y~x+x2)
betas = modlin$coefficients
plot(x, y, xlab="", ylab="", mgp=c(1.5,0.5,0), col="tomato", yaxt="n",
     xaxt="n", cex.axis=0.7, main="Ajuste cuadrático", cex.main=0.8, pch=20,
     ylim=yl, xlim=c(-0.1,3.1))
axis(1,at=0:3,cex.axis=0.8)
curve(betas[1]+betas[2]*x+betas[3]*x^2, add=T, col="purple", lwd=2)
```



```r
x2 = x^2
x3 = x^3
modlin = lm(y~x+x2+x3)
betas = modlin$coefficients
plot(x, y, xlab="", ylab="", mgp=c(1.5,0.5,0), col="tomato", yaxt="n",
     xaxt="n", cex.axis=0.7, main="Ajuste cúbico", cex.main=0.8, pch=20,
     ylim=yl, xlim=c(-0.1,3.1))
axis(1,at=0:3,cex.axis=0.8)
curve(betas[1]+betas[2]*x+betas[3]*x^2+betas[4]*x^3,
      add=T, col="purple", lwd=2)
```



```r
x2 = x^2
x3 = x^3
x4 = x^4
modlin = lm(y~x+x2+x3+x4)
betas = modlin$coefficients
plot(x, y, xlab="", ylab="", mgp=c(1.5,0.5,0), col="tomato", yaxt="n",
     xaxt="n", cex.axis=0.7, main="Ajuste cuártico", cex.main=0.8, pch=20,
     ylim=yl, xlim=c(-0.1,3.1))
axis(1,at=0:3,cex.axis=0.8)
curve(betas[1]+betas[2]*x+betas[3]*x^2+betas[4]*x^3+betas[5]*x^4,
      add=T, col="purple", lwd=2)
```




Es dificil que un fenómeno sea modelable por un polinomio en todo el rango de su dominio. Hay ocasiones en las que es más razonable pensar que en cierto rango de valores de la variable independiente el comportamiento sigue una relación polinomial y en otro rango es modelable por otro polinomio.

Ahora vamos a hacer el ajuste de un modelo lineal por pedazos

$$\mathbb{E}(y_i|x_i) = \begin{cases} a + bx_i & \text{si} \quad 0 \leq x_i \leq 1 \\ c + dx_i & \text{si} \quad 1 < x_i \leq 2 \\ e + fx_i & \text{si} \quad 2 < x_i \leq 3 \end{cases}$$

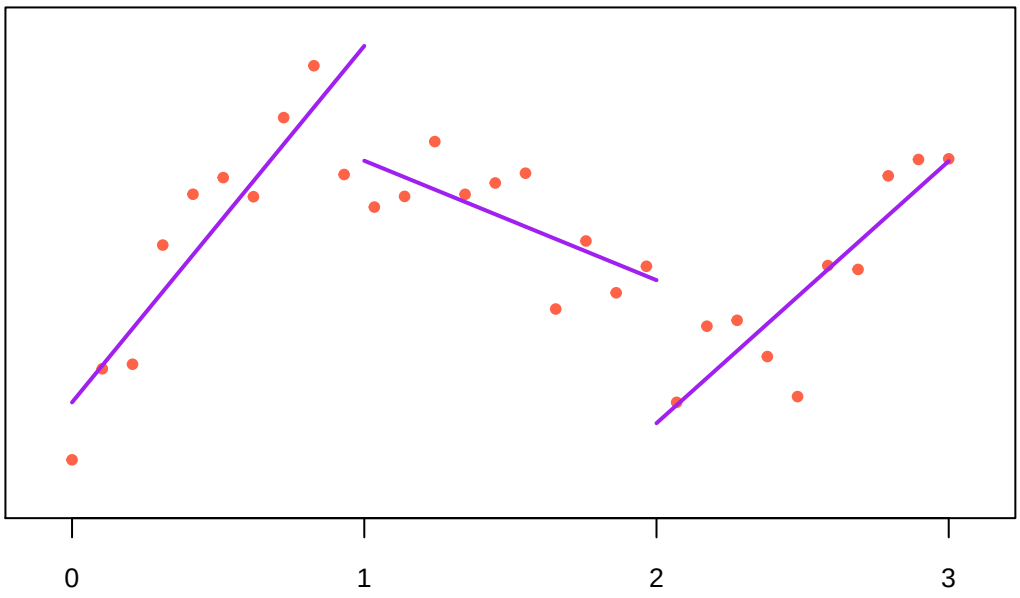
Este modelo puede ponerse en la forma $y = X\beta + e$, con

$$X = \begin{bmatrix} 1 & x_1 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_{10} & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & x_{11} & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 1 & x_{20} & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & x_{21} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 1 & x_{30} \end{bmatrix} \quad \text{y} \quad \beta = \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \end{bmatrix}$$

```
library("psych")
```

```
## Warning: package 'psych' was built under R version 3.3.3
```

```
A = matrix(c(rep(1,10), x[1:10]),  ncol=2)
B = matrix(c(rep(1,10), x[11:20]), ncol=2)
C = matrix(c(rep(1,10), x[21:30]), ncol=2)

X = superMatrix(A,B)
X = superMatrix(X,C)

bets = solve(t(X)%*%X, t(X)%*%y)

plot(x, y, xlab="", ylab="", mgp=c(1.5,0.5,0), col="tomato", yaxt="n",
     xaxt="n", cex.axis=0.7, main="Ajuste lineal por pedazos", cex.main=0.8, pch=20,
     ylim=yl, xlim=c(-0.1,3.1))
axis(1,at=0:3,cex.axis=0.8)
curve(bets[1]+bets[2]*x, from = 0, to = 1,
      add=T, col="purple", lwd=2)
curve(bets[3]+bets[4]*x, from = 1, to = 2,
      add=T, col="purple", lwd=2)
curve(bets[5]+bets[6]*x, from = 2, to = 3,
      add=T, col="purple", lwd=2)
```




Desde el punto de vista de modelación, el modelo anterior no es aceptable pues si nos acercamos a $x = 1$ por la izquierda el modelo nos predice un cierto valor $y$, pero si nos acercamos a $x = 1$ por la derecha, el modelo nos predice un valor diferente para $y$, mucho más bajo.

Lo que le falta a este modelo es "continuidad". Para hacer que las rectas se "peguen"en los puntos 1 y 2 (a estos puntos se les llama **nodos**), necesitamos que se cumplan las ecuaciones:

$$\begin{aligned} a + k_1 b &= c + k_1 d \\ c + k_2 d &= e + k_2 f \end{aligned}$$

donde $k_1$ y $k_2$ son los nodos. En otras palabras, necesitamos que los parámetros del modelo satisfagan el sistema de ecuaciones lineales (2 ecuaciones y 6 incógnitas):

$$\begin{bmatrix} 1 & k_1 & -1 & -k_1 & 0 & 0 \\ 0 & 0 & 1 & k_2 & -1 & -k_2 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \equiv \quad K^T \beta = 0$$

Entonces, para estimar los parámetros del modelo tenemos que resolver el siguiente problema de minimización sujeta a restricciones:

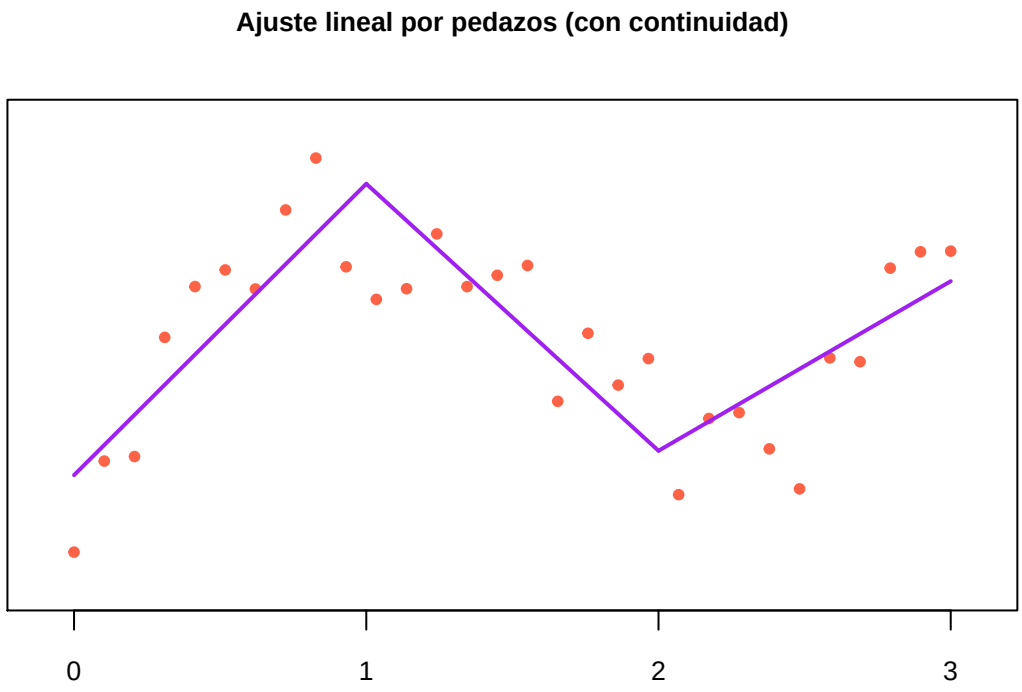
$$\min_{\beta} (y - X\beta)^T (y - X\beta) \quad \text{sujeta a} \quad K^T \beta = 0$$

La solución de este problema (vista en alguna ayudantía) puede obtenerse usando multiplicadores de Lagrange, que es:

$$\tilde{\beta} = \hat{\beta} - (X^T X)^{-1} K [K^T (X^T X)^{-1} K]^{-1} K^T \hat{\beta}$$

dinde $\hat{\beta}$ es la solución por mínimos cuadrados sin restricciones.

```
k1   = 1
k2   = 2
Kt   = matrix(c(1,k1,-1,-k1,0,0,0,0,1,k2,-1,-k2),nrow=2,byrow=T)
aux1 = solve(t(X)%*%X, t(Kt))
aux2 = solve((Kt%*%solve(t(X)%*%X))%*%t(Kt), Kt%*%bets)
tilbets = bets-aux1%*%aux2

plot(x, y, xlab="", ylab="", mgp=c(1.5,0.5,0), col="tomato", yaxt="n",
     xaxt="n", cex.axis=0.7, main="Ajuste lineal por pedazos (con continuidad)", cex.main=0.8, pch=20,
     ylim=yl, xlim=c(-0.1,3.1))
axis(1,at=0:3,cex.axis=0.8)
curve(tilbets[1]+tilbets[2]*x, from = 0, to = 1,
      add=T, col="purple", lwd=2)
curve(tilbets[3]+tilbets[4]*x, from = 1, to = 2,
      add=T, col="purple", lwd=2)
curve(tilbets[5]+tilbets[6]*x, from = 2, to = 3,
      add=T, col="purple", lwd=2)
```




Ahora podemos considerar modelos cúbicos por pedazos y que satisfagan condiciones de continuidad, es decir, queremos ajustar el modelo:

$$\mathbb{E}(y_i|x_i) = \begin{cases} a + bx_i + cx_i^2 + dx_i^3 & \text{si} \quad 0 \leq x_i \leq 1 \\ e + fx_i + gx_i^2 + hx_i^3 & \text{si} \quad 1 < x_i \leq 2 \\ i + jx_i + kx_i^2 + lx_i^3 & \text{si} \quad 2 < x_i \leq 3 \end{cases}$$

Este modelo puede ponerse en la forma $y = X\beta + e$, con

$$X = \begin{bmatrix} 1 & x_1 & x_1^2 & x_1^3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_{10} & x_{10}^2 & x_{10}^3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & x_{11} & x_{11}^2 & x_{11}^3 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 1 & x_{20} & x_{20}^2 & x_{20}^3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & x_{21} & x_{21}^2 & x_{21}^3 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & x_{30} & x_{30}^2 & x_{30}^3 \end{bmatrix} \quad \text{y} \quad \beta = \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \\ g \\ h \\ i \\ j \\ k \\ l \end{bmatrix}$$

y como queremos condiciones de continuidad, entonces las ecuaciones son:

$$\begin{aligned} a + bk_1 + ck_1^2 + dk_1^3 &= e + fk_1 + gk_1^2 + hk_1^3 \\ e + fk_2 + gk_2^2 + hk_2^3 &= i + jk_2 + kk_2^2 + lk_2^3 \end{aligned}$$

En forma matricial, estas condiciones son

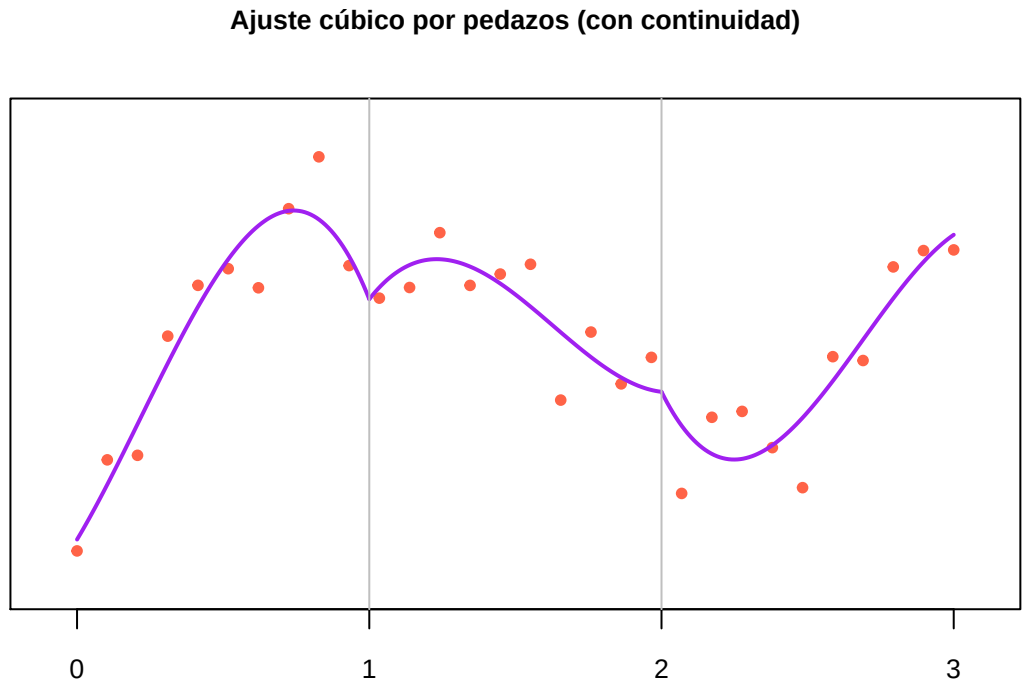
$$\begin{bmatrix} 1 & k_1 & k_1^2 & k_1^3 & -1 & -k_1 & -k_1^2 & -k_1^3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & k_2 & k_2^2 & k_2^3 & -1 & -k_2 & -k_2^2 & -k_3^3 \end{bmatrix} \beta_{12\times 1} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \equiv \quad K^T\beta = 0$$

donde $k_1$ y $k_2$ son los nodos.

Nuevamente, como en el caso anterior, lo que se quiere es estimar $\beta$ mediante aquél valor que minimice la suma de cuadrados residual, $SCE(\beta) = (y - X\beta)^T(y - X\beta)$, sujeta a las restricciones $K^T\beta = 0$.

```
# Construimos la matriz disenho
A = matrix(c(rep(1,10), x[1:10],  x[1:10]^2,  x[1:10]^3),  ncol=4)
B = matrix(c(rep(1,10), x[11:20], x[11:20]^2, x[11:20]^3), ncol=4)
C = matrix(c(rep(1,10), x[21:30], x[21:30]^2, x[21:30]^3), ncol=4)

X = superMatrix(A,B)
X = superMatrix(X,C)

# Calculamos los estimadores no restringidos
bets = solve(t(X)%*%X, t(X)%*%y)

# Calculamos los estimadores restringidos
Kt   = matrix(c(1,k1,k1^2,k1^3,-1,-k1,-k1^2,-k1^3,0,0,0,0,
                0,0,0,0,1,k2,k2^2,k2^3,-1,-k2,-k2^2,-k2^3),nrow=2,
              byrow=T)
aux1 = solve(t(X)%*%X, t(Kt))
aux2 = solve((Kt%*%solve(t(X)%*%X))%*%t(Kt), Kt%*%bets)
tilbets = bets-aux1%*%aux2

plot(x, y, xlab="", ylab="", mgp=c(1.5,0.5,0), col="tomato", yaxt="n",
```

```
     xaxt="n", cex.axis=0.7, main="Ajuste cúbico por pedazos (con continuidad)", cex.main=0.8, pch=20,
     ylim=yl, xlim=c(-0.1,3.1))
axis(1,at=0:3,cex.axis=0.8)
curve(tilbets[1]+tilbets[2]*x+tilbets[3]*x^2+tilbets[4]*x^3,
      from = 0, to = 1,
      add=T, col="purple", lwd=2)
curve(tilbets[5]+tilbets[6]*x+tilbets[7]*x^2+tilbets[8]*x^3,
      from = 1, to = 2,
      add=T, col="purple", lwd=2)
curve(tilbets[9]+tilbets[10]*x+tilbets[11]*x^2+tilbets[12]*x^3,
      from = 2, to = 3,
      add=T, col="purple", lwd=2)
abline(v=c(k1,k2),col="grey75",lwd=1)
```




Inmediatamente observamos que algo no está bien en la solución obtenida. Si bien, se tiene continuidad, a la gráfica le falta "suavidad" en los nodos; así que podemos pedir que las derivadas sean continuas ahí. La derivada de la primer cúbica es $b + 2cx + 3dx^2$, queremos que sea igual a la derivada de la segunda cúbica, $f + 2gx + 3hx^2$ cuando las evaluemos en $x = k_1$. Similarmente, en el segundo nodo, también queremos que las derivadas sean iguales. Estas observaciones se traducen en las ecuaciones:

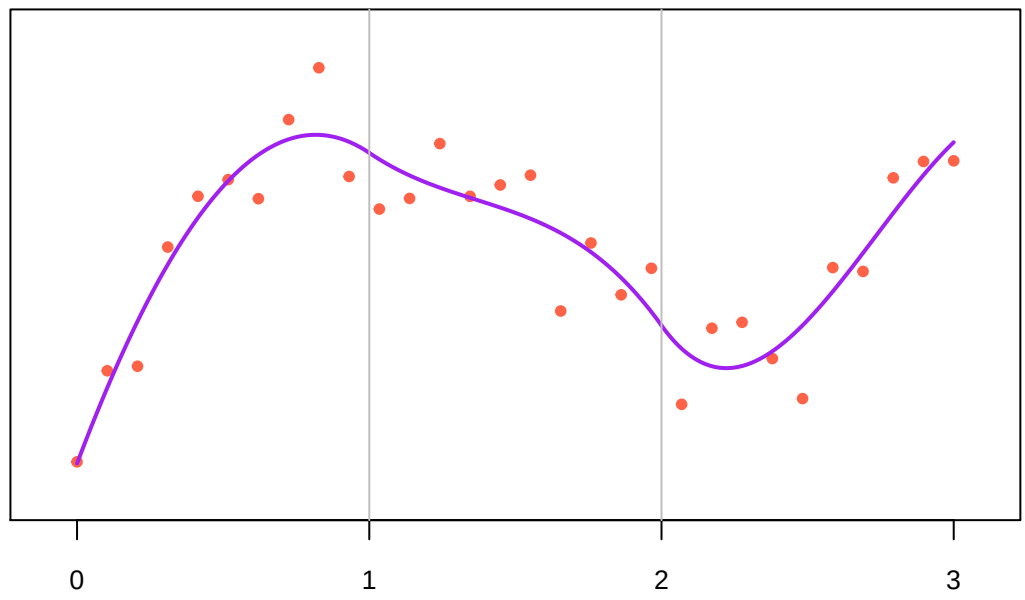
$$b + 2ck_1 + 3dk_1^2 = f + 2gk_1 + 3hk_1^2$$
$$f + 2gk_2 + 3hk_2^2 = j + 2kk_2 + 3lk_2^2$$

De modo que el total de restricciones es ahora

$$\begin{bmatrix} 1 & k_1 & k_1^2 & k_1^3 & -1 & -k_1 & -k_1^2 & -k_1^3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & k_2 & k_2^2 & k_2^3 & -1 & -k_2 & -k_2^2 & -k_3^3 \\ 0 & 1 & 2k_1 & 3k_1^3 & 0 & -1 & -2k_1 & -3k_1^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 2k_2 & 3k_2^3 & 0 & -1 & -2k_2 & -3k_2^2 \end{bmatrix} \beta_{12\times 1} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad \equiv \quad K^T\beta = 0$$

donde $k_1$ y $k_2$ son los nodos.

```
# Calculamos los estimadores restringidos
AUX =  matrix(c(0,1,2*k1,3*k1^2,0,-1,-2*k1,-3*k1^2,0,0,0,0,
                0,0,0,0,0,1,2*k2,3*k2^2,0,-1,-2*k2,-3*k2^2),nrow=2,
              byrow=T)

Kt = rbind(Kt, AUX)

aux1 = solve(t(X)%*%X, t(Kt))
aux2 = solve((Kt%*%solve(t(X)%*%X))%*%t(Kt), Kt%*%bets)
tilbets = bets-aux1%*%aux2

plot(x, y, xlab="", ylab="", mgp=c(1.5,0.5,0), col="tomato", yaxt="n",
     xaxt="n", cex.axis=0.7, main="Ajuste cúbico con continuidad en primeras derivadas", cex.main=0.8, p
     ylim=yl, xlim=c(-0.1,3.1))
axis(1,at=0:3,cex.axis=0.8)
curve(tilbets[1]+tilbets[2]*x+tilbets[3]*x^2+tilbets[4]*x^3,
      from = 0, to = 1,
      add=T, col="purple", lwd=2)
curve(tilbets[5]+tilbets[6]*x+tilbets[7]*x^2+tilbets[8]*x^3,
      from = 1, to = 2,
      add=T, col="purple", lwd=2)
curve(tilbets[9]+tilbets[10]*x+tilbets[11]*x^2+tilbets[12]*x^3,
      from = 2, to = 3,
      add=T, col="purple", lwd=2)
abline(v=c(k1,k2),col="grey75",lwd=1)
```



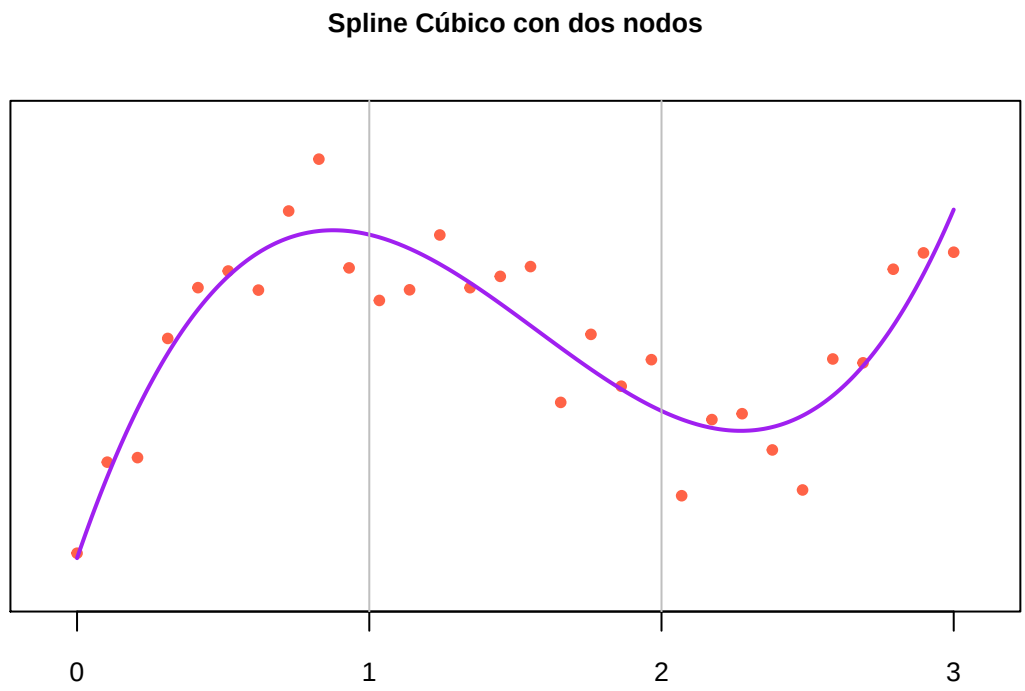

Una petición más. Queremos agregar continuidad en la segunda derivada, esto hará que el modelo no sea tan 'curvo'. En otras palabras:

$$2c + 6dk_1 = 2g + 6hk_1$$
$$2g + 6hk_2 = 2k + 6lk_2$$

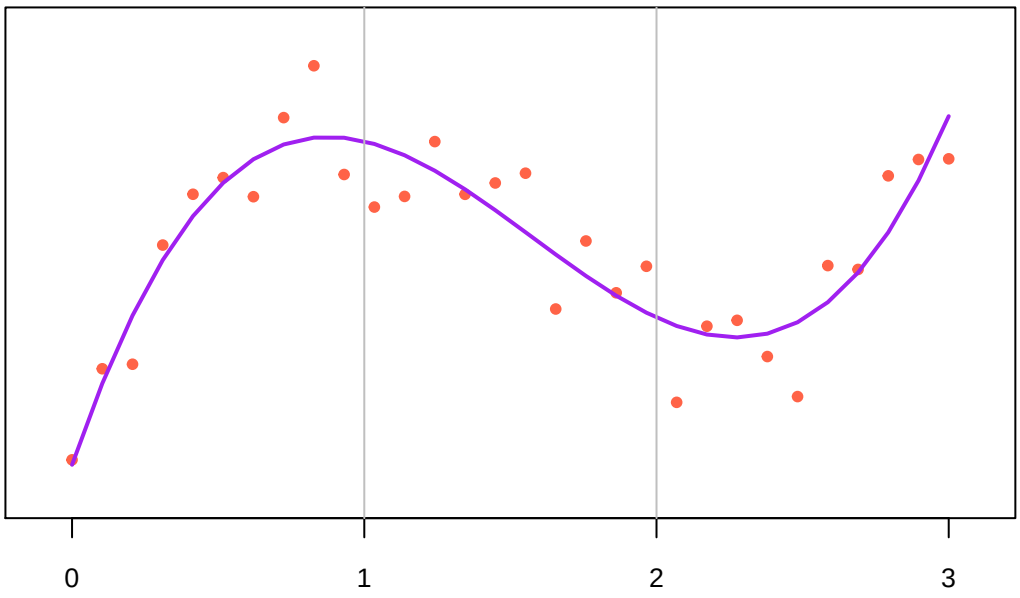
Agregando estas restricciones a las enteriores se obtiene el ajuste llamado **Spline Cúbico**.

```
# Calculamos los estimadores restringidos
AUX =  matrix(c(0,0,2,6*k1,0,0,-2,-6*k1,0,0,0,0,
                0,0,0,0,0,0,2,6*k2,0,0,-2,-6*k2),nrow=2,
              byrow=T)

Kt = rbind(Kt, AUX)

aux1 = solve(t(X)%*%X, t(Kt))
aux2 = solve((Kt%*%solve(t(X)%*%X))%*%t(Kt), Kt%*%bets)
tilbets = bets-aux1%*%aux2

plot(x, y, xlab="", ylab="", mgp=c(1.5,0.5,0), col="tomato", yaxt="n",
     xaxt="n", cex.axis=0.7, main="Spline Cúbico con dos nodos", cex.main=0.8, pch=20,
     ylim=yl, xlim=c(-0.1,3.1))
axis(1,at=0:3,cex.axis=0.8)
curve(tilbets[1]+tilbets[2]*x+tilbets[3]*x^2+tilbets[4]*x^3,
      from = 0, to = 1,
      add=T, col="purple", lwd=2)
curve(tilbets[5]+tilbets[6]*x+tilbets[7]*x^2+tilbets[8]*x^3,
```

```
      from = 1, to = 2,
      add=T, col="purple", lwd=2)
curve(tilbets[9]+tilbets[10]*x+tilbets[11]*x^2+tilbets[12]*x^3,
      from = 2, to = 3,
      add=T, col="purple", lwd=2)
abline(v=c(k1,k2),col="grey75",lwd=1)
```




```
# Haciendo Splines Cubicos, usando paquete "splines"
library("splines")
splines = bs(x, knots = c(k1,k2))
model <- lm(y~splines)
plot(x, y, xlab="", ylab="", mgp=c(1.5,0.5,0), col="tomato", yaxt="n",
     xaxt="n", cex.axis=0.7, main="B-Spline Cúbico con dos nodos", cex.main=0.8, pch=20,
     ylim=yl, xlim=c(-0.1,3.1))
axis(1,at=0:3,cex.axis=0.8)
lines(x,predict(model), col="purple", lwd=2)
abline(v=c(k1,k2),col="grey75",lwd=1)
```


B−Spline Cúbico con dos nodos
0
1
2
3